Assignment 2 INF367

Johanna Jøsang

September 20, 2020

1 PAC learning

1.1 Show that a CNF is PAC learnable in polynomial time with classification noise

Let the learning framework F and its components be defined as in the assignment description.

Upon encountering classification noise, the main problem with the old algorithm for CNF learning, is that if it recieves a misclassified negative example, it can make irreversible deletions of literals that actually should have stayed in the hypothesis. Since this error arises due to the algorithm basing its learning on a single example, it makes sense for an algorithm receiving noisy examples to base its learning on statistics. We will now look at such an algorithm, which samples literals individually and adds those that disagree least often on positive examples to the hypothesis.

1.1.1 Categorizing literals as *significant* and *harmful*

Lets denote $p_0(z)$ as the probability of the literal z violating an evaluation selected from the distribution D. If $p_0(z)$ is very small then it is not unsafe to include it in the hypothesis, since most of the time does not violate a valuation drawn from D. z is considered an unimportant literal. A literal z is considered *significant* if $p_0(z) \geq \epsilon/(16(2n)^2)$.
Let C denote the concept class of conjunctions of literals, and $c \in C$ be a target. $p_{01}(z)$ is the probability of a literal z violating an evaluation selected from the distribution which the noisy oracle labels as positive. If $p_{01}(z)$ is large then there is a high chance that it could cause the hypothesis h to misclassify if z is included in h. Hence, a literal z is considered *harmful* if $p_{01}(z) \geq \epsilon/(2(2n))$. Since $p_{01}(z) \leq p_0(z)$, every harmful literal is also significant.

1.1.2 A hypothesis with all significant, but non-harmful literals has a true error less than ϵ

The goal then is for h to contain all the significant, but non-harmful literals. We can show that this ensures that $error(h, c, D) < \epsilon$.
Let us start by examining the two cases where h might misclassify. For some valuation $a \in D$:

1. **$c(a) = 0$ and $h(a) = 1$.** This event occurs only if there is some literal z present in c but not in h, and $a(z) = 0$. Since we stated that h contained all the significant, yet non-harmful literals, and c doesn't contain harmful literals by the definition of harmful, any such a literal z must not be significant.

   This means that $D(c(a) = 0 \wedge h(a) = 1)$ is less than or equal to the probability that some insignificant literal z is in a valuated to 0 in a.

   $$D(c(a) = 0 \wedge h(a) = 1) \leq \sum\nolimits_{z \in (h-c)} p_0(z), \text{ where } p_0(z) < \epsilon/(16(2n)^2)$$

   Since this could be at most $2n$ such literals, by union bound this probability is $2n \times (\epsilon/(16(2n)^2))$. So:[1]

   $$D(c(a) = 0 \wedge h(a) = 1) \leq 2n \times (\epsilon/(16(2n)^2)) = \epsilon/(16(2n)) \leq \epsilon/32 < \epsilon/2$$

2. **$c(a) = 1$ and $h(a) = 0$.** This event occurs only if there is some literal z present in h but not in c, and $a(z) = 0$. Since h contains no harmful literals, $D(c(a) = 0 \wedge h(a) = 1)$ is bounded by the probability that for some literal z, $a(z) = 0$ and $c(a) = 1$.

   $$D(c(a) = 0 \wedge h(a) = 1) \leq \sum\nolimits_{z \in (h-c)} p_{01}(z), \text{ where } p_{01}(z) < \epsilon/(2(2n))$$

   [2] As there could be at most $2n$ such literals, by union bound this probability is $2n \times (\epsilon/2(2n))$. So:

   $$D(c(a) = 0 \wedge h(a) = 1) \leq 2n \times \epsilon/(2(2n)) = \epsilon/2$$

Thus, if the algorithm manages to construct a hypothesis h containing all the significant literals that are not harmful then:

$$error(h, c, D) \leq \epsilon/2 + \epsilon/2 = \epsilon$$

[1] $\epsilon/(16(2n)) \leq \epsilon/32$ because $n \geq 1$
[2] $h - c$ denotes the set of clauses in h but not in c

1.1.3 Description of the algorithm for learning in the presence of noise

We denote the algorithms training set as S, with $|S| = k$. S has a noise rate of η_b. ϵ is the accepted error rate of a hypothesis produced by the algorithm. ϵ, k and η_b are given as parameters to the algorithm.[3]
The algorithm so far has no information about $p_0(z)$ or $p_{01}(z)$, but can obtain estimates for these through looking at the examples in S. The estimate for $p_0(z)$ is calculated by dividing the number of valuations violated by z to the total number of valuations. With this the algorithm can be ensured to contain all the significant literals. We denote the set that, with high probability, contains all the significant literals as I.

$$I = \{z \in Z, p_0(z) \geq \epsilon/(32(2n)^2)\}$$

Note that since I am using $p_0(z)$, not the number of times z violates an example, I do not divide $p_0(z)$ by k in the inequality above. Now it remains to identify all the harmful literals in I. Let $p_{0+}(z)$ be the probability that a valuation that is categorized as positive in S, will be violated by z.

$$p_{0+}(z) = D((a, +) \in O_\eta, a(z) = 0)$$

This event can occur with or without reporting error:

- Without reporting error: $a(z) = 0$ and $a(c) = 1$
- With reporting error: $a(z) = 0$ and $a(c) = 0$

The proportion of samples with reporting error will be denoted by $apr(z)$.

$$apr(z) = p_{0+}(z)/p_0(z)$$

Now we would like to know how high $apr(z)$ can be before z is rejected. From calculations made in section 2.2 "Learning from Noisy Examples" by Angluin et al. we know that there is a separation of at least $s = \frac{\epsilon}{2(2n)}(1-2\eta)$ in the expected difference between the quantity of literals in h that are harmful, and clauses in h that are important. η is the true noise rate. Let η_b be an upperbound for η, such that

$$\eta \leq \eta_b \leq \frac{1}{2}$$

[3]In our description of the algorithm we allowed k to be a constant determined by the user, but in "Learning from Noisy Examples" a spesific bound for the mininum value of k is given. Theorem 2 in section 2.2 states that if we draw a sequence S of k examples from an orcale of which the noise rate is upper bounded by η_b, and has a distribution D, where

$$k \geq \frac{2}{\epsilon^2(1-2\eta_b)^2} \ln\left(\frac{2(2n+1)}{\delta}\right)$$

then

$$D^k(error(h_S, c, D) \leq \epsilon) \leq \delta$$

Which allows us to write:

$$s \geq s_b = \frac{\epsilon}{2(2n)}(1 - 2\eta_b)$$

So the algorithm finds an estimate η' for η, and determines that

$$apr(z) \leq \eta' + \frac{s_b}{2}$$

is the set of literals in I that are not harmful. This set is the hypothesis which the algorithm creates.
For the estimate for the noise, η', there are two candidates:

1. $\eta_1 = p_-/k$, where p_- is the number of negatively classified examples
2. $\eta_1 = \min\{apr(z), z \in I\}$

Of which $\eta' = \min\{\eta_1, \eta_2\}$.
The final output of the algorithm then is the set defined as:

$$\{z \mid z \in I, apr(z) \leq \eta' + \frac{\epsilon(1 - 2\eta_b)}{8n}\}$$

1.1.4 Runtime analysis

The algorithm needs to calculate

- p_-, runtime k
- p_0 for all z, runtime $k(2n)$
- I, runtime $2n$
- p_{01} for all z, runtime $k(2n)$
- $apr(z)$ for all z, runtime $2n$
- η', worst case runtime $2n + 2$
- $\{z \mid z \in I, apr(z) \leq \eta' + \frac{\epsilon(1-2\eta_b)}{8n}\}$, worst case runtime $2n$

So we can see that the algorithm has a big-O of $O(k(2n))$, and hence runs in polynomial time.

1.1.5 Proof of correctness of the algorithm

Proof of the correctness of this algorithm can be found in section 3.3 in "Learning from Noisy Examples" by Angluin et. al. The proof examines the ways in which the algorithm may go astray, and shows that the event of these occurring have a probability of at most δ. Furthermore it refers to an earlier proof in the paper, which we went through in section 1.1.2 in this assignment, showing that such a hypothesis which this algorithm produces has a true error less than ϵ.

Hence we can conclude that the algorithm fulfills the criteria for being PAC learnable in polynomial time.

1.2 Upper bound for VC dimension of F

We have the theorem that $VC(F) \leq log_2(|H|)$ if H is finite. For the learning framework F, H is the set of all conjunctions of literals that can be formulated in the form $V = v_1...v_n$. Since n is a positive integer, there is a finite number of possible valuations over the set V. Hence H is finite, and $VC(F) \leq log_2(|H|)$.